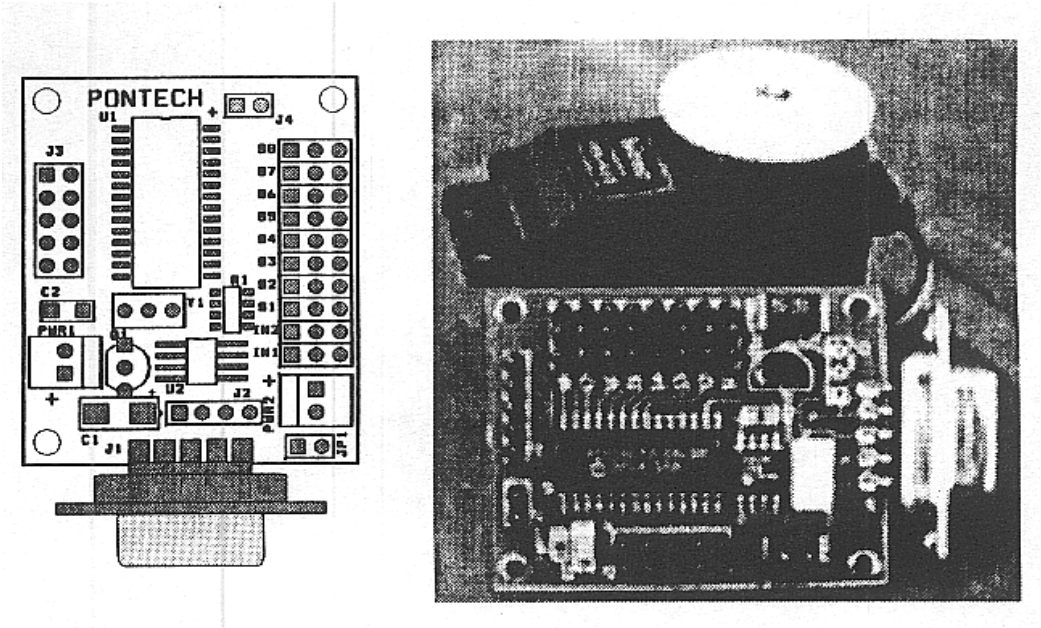
Jameco Part number 153648

SV203/300 - **Servo Motor Controller Board**



The SV203 and SV300 is a PIC16C73 microcontroller based servo motor controller board that accepts RS-232 serial data from a host computer and outputs PWM (pulse-width-modulated) signals to control up to eight RC (radio-controlled, i.e. model airplanes, cars, etc.) servo motors. Unused servo pins can be reconfigured as digital outputs for controlling on/off devices. LED displays can be driven directly by the pins but devices such as relays and solenoids may need a simple transistor driver circuit.

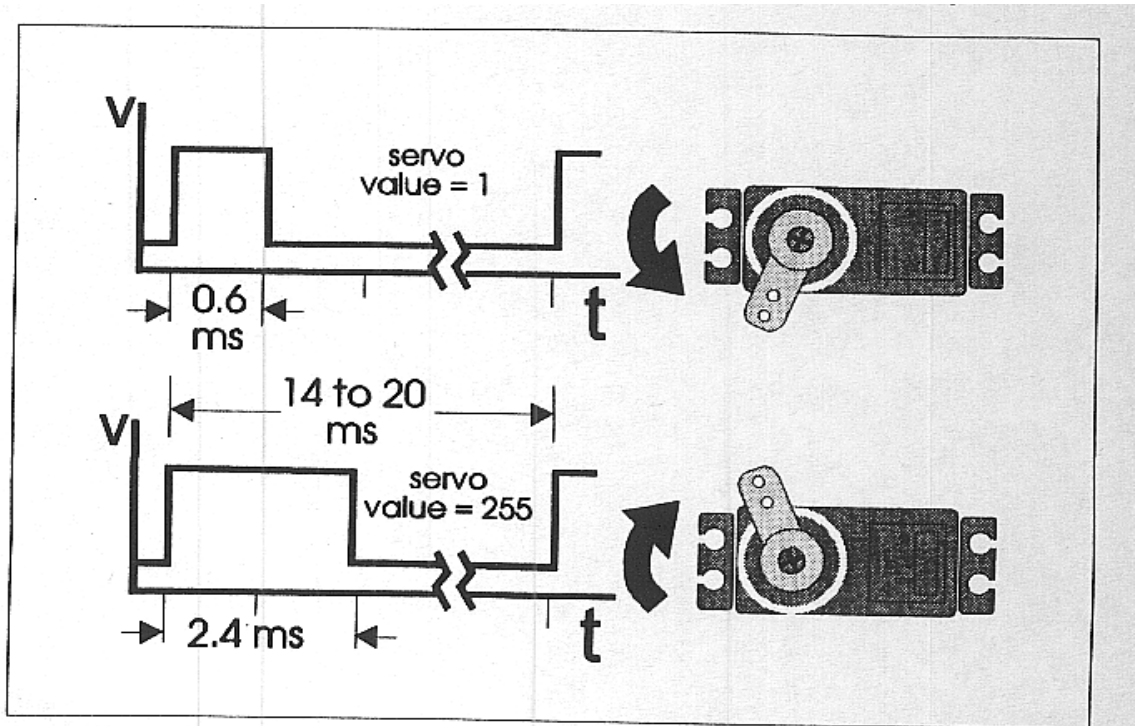
There's also a five channel A/D port for reading analog voltage between 0-5 volts, and a SPI (synchronous peripheral interface) port which allow data to be shifted in or out serially.

The SV300 has the added feature of being able to run a standalone BASIC program stored on board an 8K EEPROM. An option IR-Receiver is also available to be added to the SV300 to allow it to receive command from a Infrared Remote controller.

SV203/300 - **Servo Basics**

RC servos operate using feedback to compare the current position to an input pulse width, which typically repeats every 14 to 20 ms (milliseconds). If the pulse width lasts for approximately 0.6 ms, the servo will rotate to a maximum position. If the pulse width is increased to approximately 2.4 ms, the servo will rotate to the opposite maximum position (figure 1). A 1.5 ms pulse will set the servo in the middle (neutral) position.

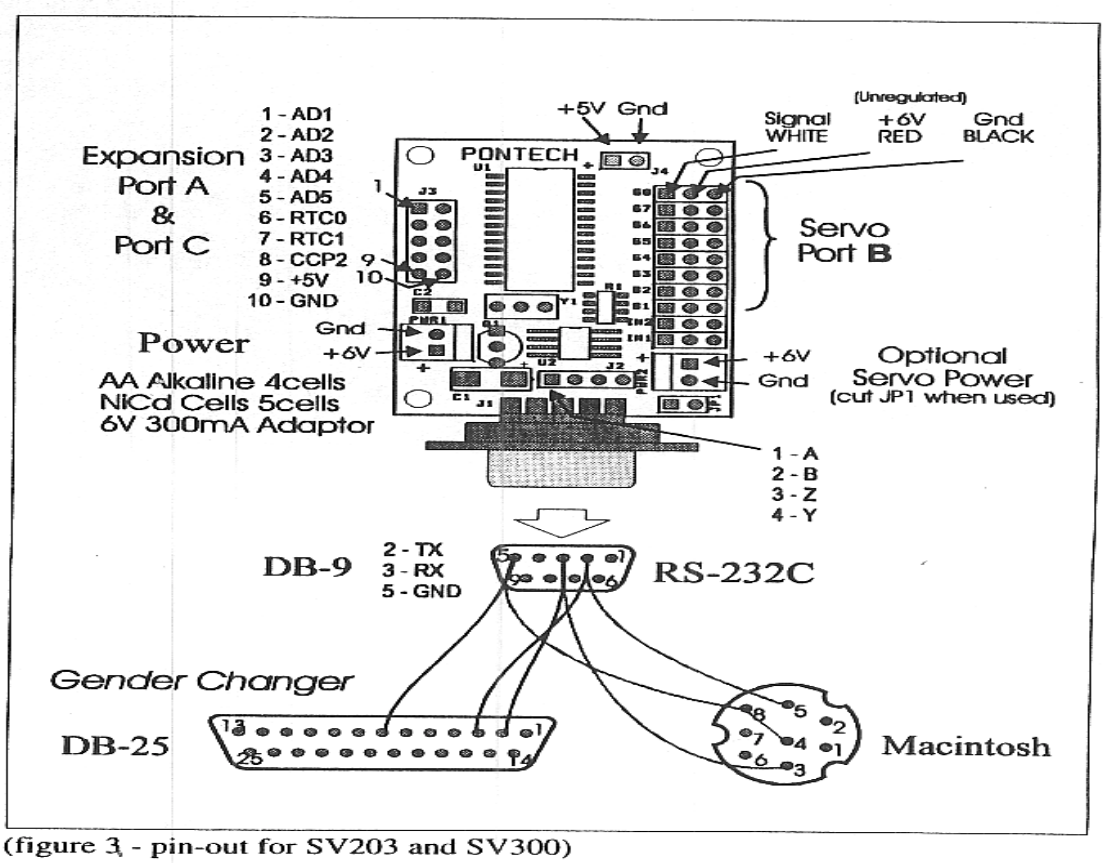
The SV203 and SV300 controller is designed to the specifications of a Futaba servo model FP-S148. These servos have a neutral position at 1.52 ms. -90 degrees at 0.6 ms, and 90 degrees at 2.4 ms. Other servos may have slightly different values for these positions.



(figure 1 - servo timing diagram)

SV203/300 - Power Supply and Pin-out

A 6 Volt DC source powers the board, either from 4 alkaline batteries or 5 NiCad cells. An AC adapter can also be used: 6VDC, at 300mA. If using NiCads, a 4-cell pack might be easier to find than a 5-cell pack. The board will operate fine with 4 cells, but may not last as long as 5 cells.

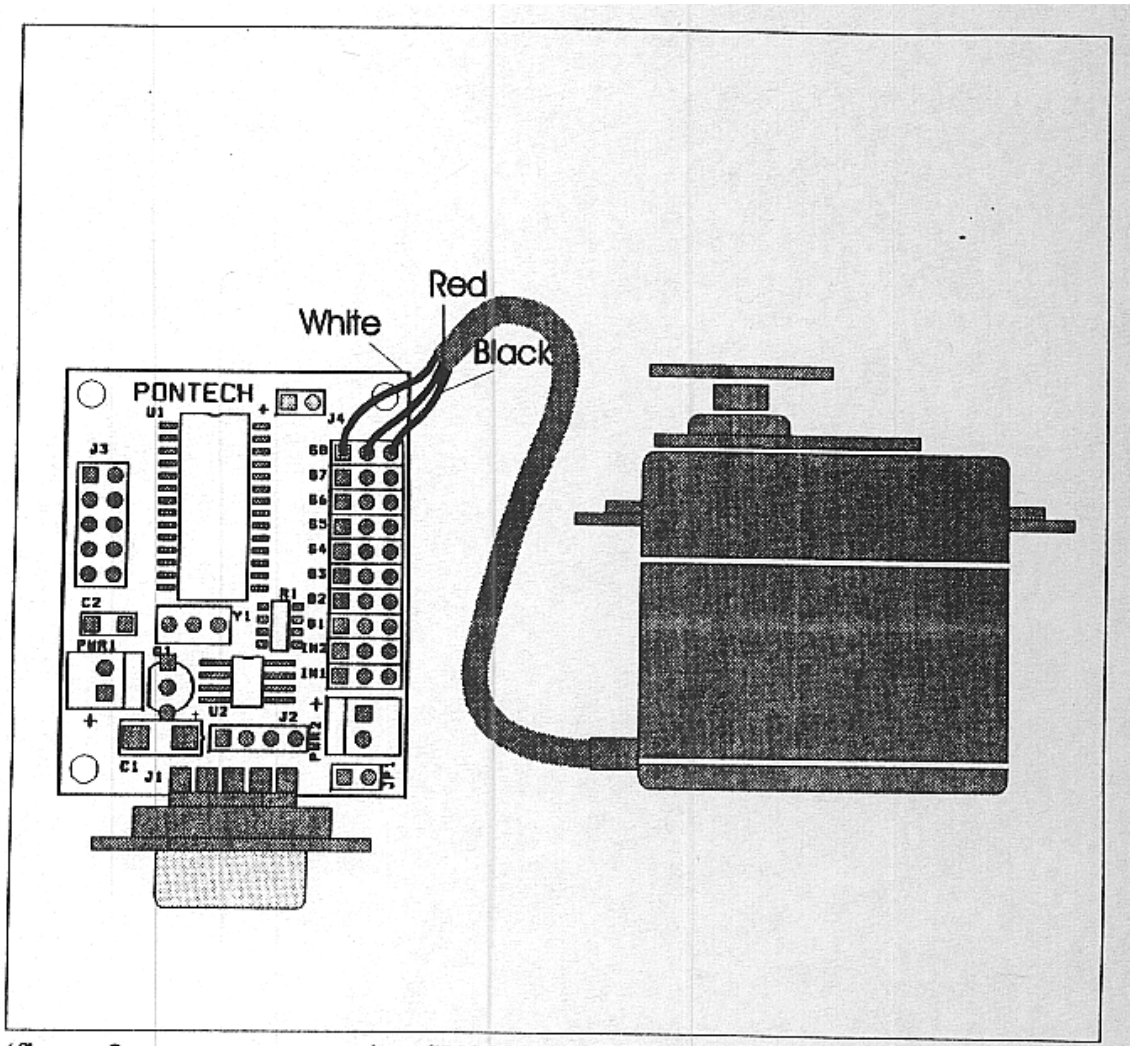


(figure 3 - pin-out for SV203 and SV300)

SV203/300 - **Servo Connections**

The servo port connectors use a 3-pin male sip (single inline pin) connector (0.1-inch spacing). The servo connector was designed to be used with Futaba type servos with J-type connectors. The servos have three colored wires, Black for ground, Red for power, and White for signal.

Connect the cable with the White wire pointing to S1...S8, and the Black wire toward the outside edge of the board.



(figure 2 - servo connected to SV203 and SV300)

```
OPEN "COM1:9600,N,8,1,CD0,CS0,DS0" FOR OUTPUT AS #1 DO

        PRINT:PRINT "*****SV203/300 Servo Controller *****"
        INPUT "Enter Board ID Number:
        INPUT "Enter Servo # to control: Servo$
        INPUT "Enter Position of Servo: Pos$


        PRINT #1, "BD";ID$;"SV";Servo$;"M";Pos$
        INPUT "Quit (y/n)"; Q$: IF Q$ = "Y" THEN EXIT DO LOOP
```

(listing 1 - Sample interface program using QBASIC)

```
* * * * *SV203/300 Servo Controller * * * * * "
Enter Board ID Number:? 1              Board width ID = 1
Enter Servo # to control:? 2           move servo #2 to position 200
Enter Position of Servo:? 200
Quit (y/n):? N
* * * * * SV203/300 Servo Controller * * * * *"
Enter Board ID Number:? 0              Any Board regardless of ID, move Servo #4 to position
                                       254
Enter Servo # to control:? 4
Enter Position of Servo:? 254
Quit (y/n):? N
* * * * * SV203/300 Servo Controller * * * * *"
Enter Board ID Number:? 1              Board with ID = 1
Enter Servo # to control:? 2           Turn off Servo #2
Enter Position of Servo:? 0
Quit (y/n):? n
* * * * * SV203/300 Servo Controller * * * * *"
Enter Board ID Number:? 1              This command is invalid because there is no Servo #9
                                       and also 256 is out of the range of the servo position.
Enter Servo # to control:? 9
Enter Position of Servo:? 256
Quit (y/n):? Y
```

(listing 2 - Example screen of when program in listing 1 is running)

Listing 1 is a simple program written in QBASIC that requests the user to input a board ID number, a servo number to control, and the position of the servo.

SV203/300 - **Operation of Board**

Connect the servos to the board, plug in the RS-232 cable to a COM port of a PC and the other end to the SV203/300. Power the board and the servos should return to the neutral position. Run the example programs.

The SV203/300 processes information one ASCII string command at a time. Each command string follows the format:

L n L n ... <enter> (maximum of 20 characters/line)

Where L is the command letter(s) in caps, n is a decimal integer number(s), and <enter> is ASCII 13. Please refer to the back of the manual for a complete command listing.

For example, the command to select a board, select a servo and move to a position are BD, SV, and M, respectively. If your want to move servo number 3 of a board with an ID number equal to 1 to position 85, you would send the flowing command string.

BD1SV3M85 <enter>

The commands can also be separated by spaces for commas for ease of reading.

BD1 SV3 M85 <enter> or
BD1, SV3, M85 <enter> or
BD 1 SV 3 M 85 <enter>

A terminal program may be used to test the functions of the board. The default setting on the board is 9600 baud, N81, with echo off. (A program called TERM.EXE is provided as a simple terminal like software to test the SV203/300 functions).

SV203/300 - **Operation of Board (continued)**

Once a board or servo is selected, it will stay selected until power is removed or another select command is received. For example, if the following commands were sent:

BD1 SV2 M100 <enter>

The next command will still move servo 2:

M150 <enter>

More then one servo can be moved at the same time in one line of a command string, just make sure you don't exceed the 20 characters per line limit, including spaces and commas:

BD1 SV1 M30 SV2 M104 SV3 M25 <enter>

The commands above will select Board #1, Move servo #1 to position 30, then move servo #2 to position 104 and move servo #3 to position 25 all at about the same time.

Any command or parameter value not in the range of the command will be ignored.

The board will start processing the command string when it sees the <enter> or ASCII 13 character. The host computer talking to the board should insert a delay of about 3 milliseconds between each command string to give time to process the commands.

SV203/300 - Commands Descriptions

BDn Select Board

Before the board will accept any commands, it must first be enabled. To enable the board, you must send the BD command followed by the board ID number. The default ID number of the board is 1. So simply send the following to enable the board:

BD1<enter>

The board ID number can be user redefined by using the WE command. This allows multiple boards of different ID number to be connected to the same serial port.

You can enable the board in two other ways: You can pre-enable the board at power-up by changing the default-settings. (see changing default-settings); or you can enable the board by sending an ID number 0, such as:

BDO <enter>

This will override the ID number checking and any boards connected to the port will be enabled regardless of the ID number of the board.

SVn Select Servo

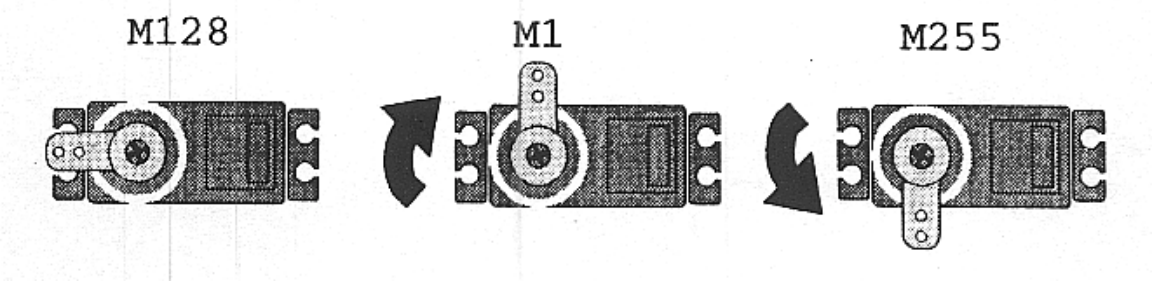
On power-up, servo #1 is pre-selected. To select another servo or to make sure the servo is selected, send "SV" followed by the servo number. The servo number must be between 1 and 8 because there's a maximum of eight servos that the board can control.

i.e. SV3 <enter>

servo number 3 will be selected. Any move commands that follow will operate on servo number 3

Mn **Move to absolute position**

This command will move the selected servo to an absolute position. The range of the position is between 1 and 255. On an RC type servo, the maximum mechanical movement is about 180 degrees. The 1 to 255 position ranges gives a precision of a little under one degree. The figure below shows what position the servo will be in given the value of position.



Sending a zero to the servo can turn off the servo:

M0 <enter>

In **Incremental move relative to current position**

This command will move the servo relative to the current position of the servo by adding or subtracting the value entered to the current servo position.

i.*e*. M100 <enter>
I10 <enter>
I-20 <enter>

The selected servo will first move to position 100, then to position 110 (100+10), and then finally to position 90 ((100+10) - 20).

SV203/300 - **Commands Descriptions**

Dn **Delay/Pause in milliseconds (ms)**

Delay commands may be added to the string to pause between movements:

SV1 M20 D1000 M100 <enter>

Servo #1 will move to position 20. There is a one-second (1000 ms) pause, and then servo #1 will move to position 100.

Caution: When using the Delay command, the board will not receive during the delay state. The host computer that's talking to the board has to wait at least the same amount of time before another command string can be sent. Any commands sent during the delay will be ignored.

PSn **Pin Set**
PCn **Pin Clear**
PTn **Pin Toggle**

These commands allow you to use the servo port as a digital output by setting, clearing or toggling individual bits of the servo port. In order to use the port as a digital output, the servo PWM must first be turned off by sending a MO command to each servo pin that you want to use as digital output.

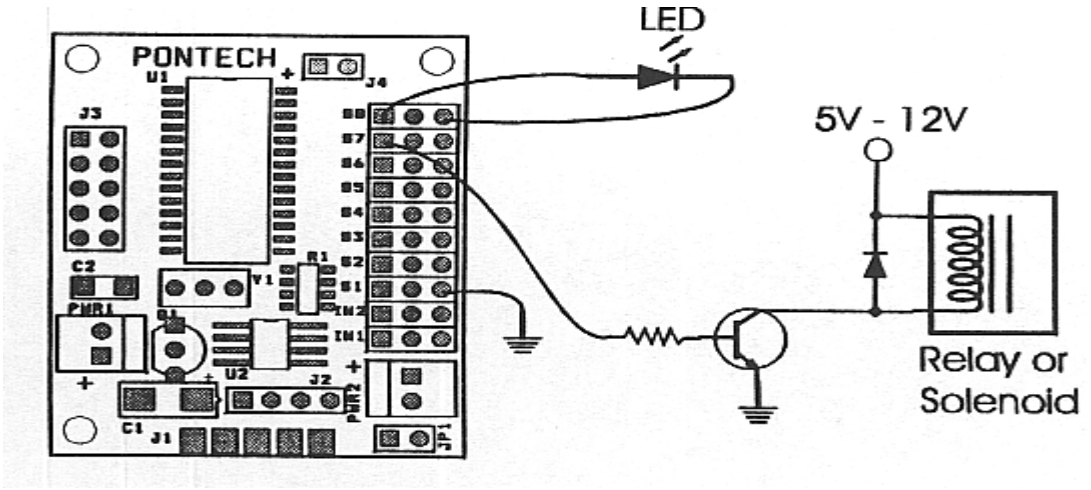
i.*e*. SV7 M0 PS7 <enter>
Pin S7 of the servo port will be set high (5 Volts).

i.*e*. SV8 M0 <enter>
PT8 <enter>

Pin S8 of the servo port will be toggled/flipped (set high if pin was low, or cleared if pin was high).

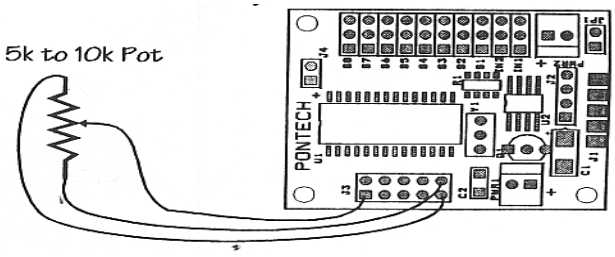
SV203/300 - **Commands Descriptions**

The pins can drive and sink up to 25 ma, a driver circuit such as the one below may be required to drive anything that uses more current such as a relay or a solenoid.



ADn **Read a voltage on the A/D port** Port A is a 5 Channel, 8-bit A/D input that can read an analog voltage. The n is a number between 1 to 5, which tells what pin on the A/D port to request. When the board receives this command, it will read the voltage on the pin specified and return a value between 0 to 255, which represents a voltage between 0 to 5 Volts.
i.e. AD1 <enter>

If wires were connected as in the figure below and the pot as in the middle position, the board would return a value close to 128 followed by <ASCII 13>, which is about 2.5 Volts.



SV203/300 - **Commands Descriptions**

SI, SOn **Shift byte In/Out the SPI port**

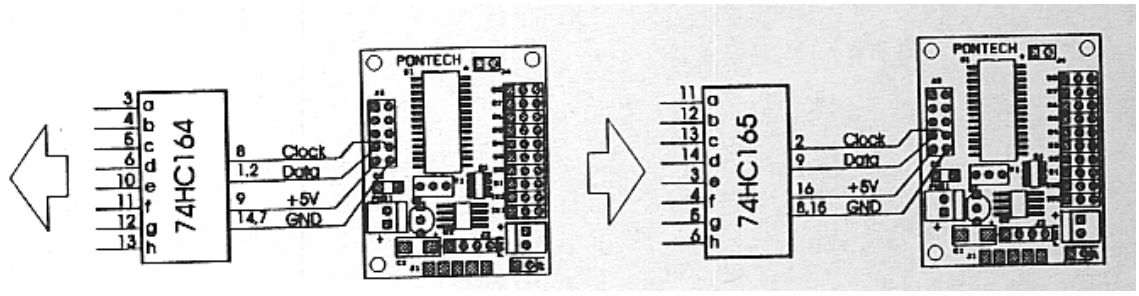
These commands allow you to use port C on the board as a synchronous serial port to shift in or out a byte of data. The figure below shows how to connect shift register chips (74HC164 or 74HC165) the to board to make a serial-to-parallel or parallel-to-serial converter.

i. *e*. SO3 <enter>

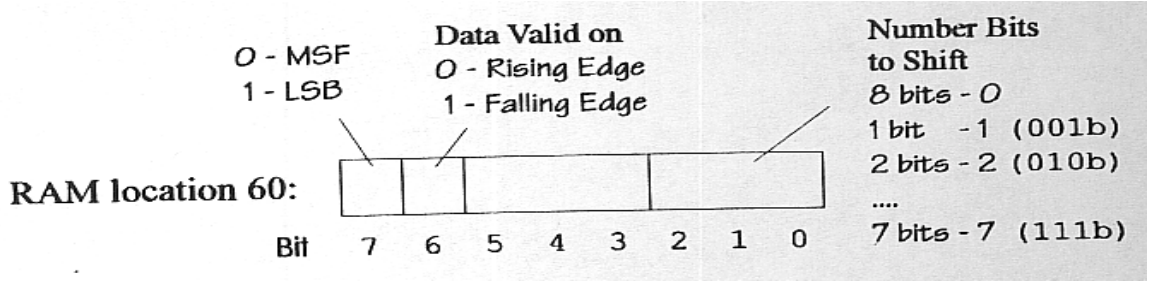
pins 3 and 4 of the '164 will be set high, while all others are low (3 dec-> 00000011 bin).

i.*e*. SI <enter>

the board will return a number between 0 to 255.



There is a configuration register at location 60 in RAM. The value of the register is initialized by the contents of EEPROM in location 11. The register configures the shift function for MSF (most significant first) or LSF (least significant first), data valid on clock going high or low, and the number of bits to shift in/out.



SV203/300 - **Commands Descriptions**

WRm n **Write to Ram**
RRm **Read from Ram**

These commands allow you to modify and read the contents of the internal register or RAM of the processor. The internal RAM is volatile memory storage, so when power is removed the contents will be erased.

i.e. WR51 20 <enter>
servo #1 will move to position 20, this command is equivalent to SV1 M20 <enter>

i.e. RR52 <enter>
the position of servo number 2 will be returned.

RAM Memory Map:

Address (m)	Usage	Note
5	Port A	
6	Port B	
7	Port C	
14	TMR1L	
15	TMR1H	
16	T1CON	
27	CCPR2L	
28	CCPR2H	
29	CCP2CON	
51 to 58	**Current Servo Position**	
59	**Servo Select**	
60	**Shift Config Register**	
133	TRIS A	
134	TRIS B	
135	TRIS C	

Note: All other RAM locations not listed are used by the system and should not be used..

SV203/300 - Commands Descriptions

WEm n **Write to EEPROM**
REm **Read from EEPROM**

These commands allow you to modify and read the contents of the external EEPROM connected to the processor. The EEPROM is a non-volatile memory storage, so any information written to it will stay even when power is removed.

i.*e.* WE0 2 <enter>
change the board ID number to #2

i.*e.* RE1 <enter>
the SV203/300 returns the initial servo value of servo #1

EEPROM Memory Map:

Address (m)	Usage	Factory default	Note
0	Board ID #	1	
1	Initial Servo #1 Value	128	0 - off or Digital
2	Initial Servo #2 Value	128	0 - off or Digital
3	Initial Servo #3 Value	128	0 - off or Digital
4	Initial Servo #4 Value	128	0 - off or Digital
5	Initial Servo #5 Value	128	0 - off or Digital
6	Initial Servo #6 Value	128	0 - off or Digital
7	Initial Servo #7 Value	128	0 - off or Digital
8	Initial Servo #8 Value	128	0 - off or Digital
9	Baud Rate	50 (9600 baud)	24 (19200 baud) 100(4800 baud) 200(2400 baud)
10	Pre Enable Flag	1	1-Yes 0-No
11	Shift Config Register	0	MSB, valid on Rising, 8-bit

SV203/300 - **Changing Default Settings**

When the board is powered-up, it first reads the external EEPROM to get its Board ID number, Initial servo positions, baud rate, and some other initial flags. These initial settings can be changed by using the WE command.

To Change ID:

WE0 3 <enter>
#3 is the new ID number

To have the board be pre-enable on power-up:

WE10 1 <enter>

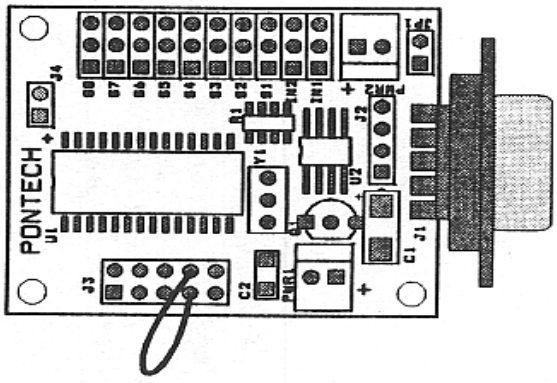
To Change Servo initialization:

WE2 0 <enter>
Servo #2 will be off when the board is turned on.

See the Memory Map on previous page for other default settings.

The board can be reset to it's factory default settings by tying pins 8 and 10 of port C together, power-up the board, and wait for about 3 seconds.

You can also use the enclosed program on disk called DEFAULT.EXE to change the default settings of the board.

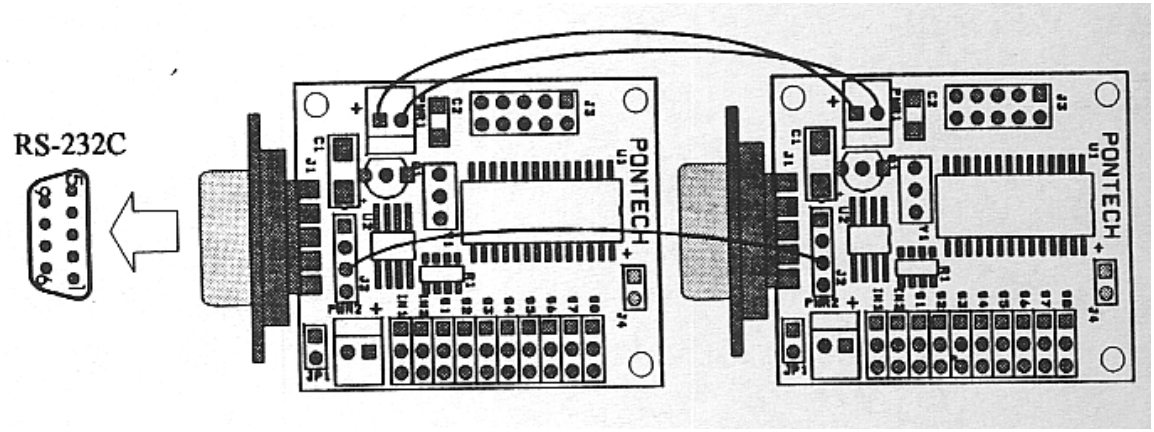


- Turn off board
- Tie Pin 7 and 8 together
- Turn on board
- wait 3 seconds
  The board will be restored with its factory default.

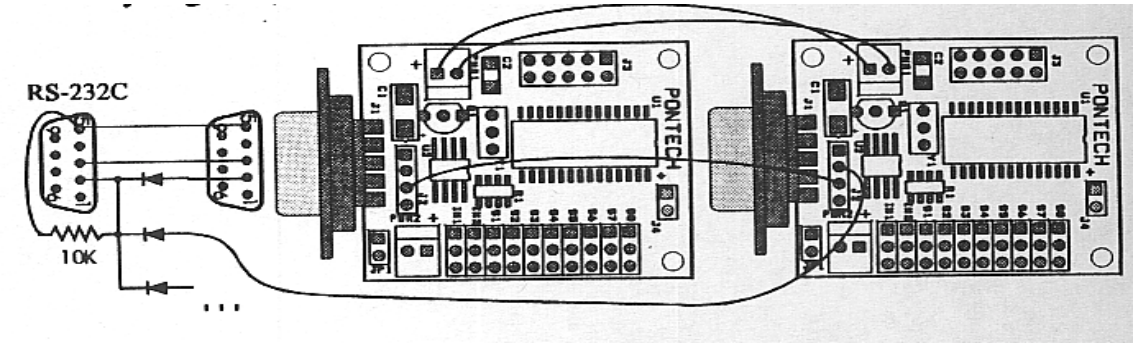
SV203/300 - **Controlling multiple boards**

You can connect multiple boards to the same serial line by reassigning different ID numbers to each board.

If you are using the second board for output only, the wiring below will work just fine.



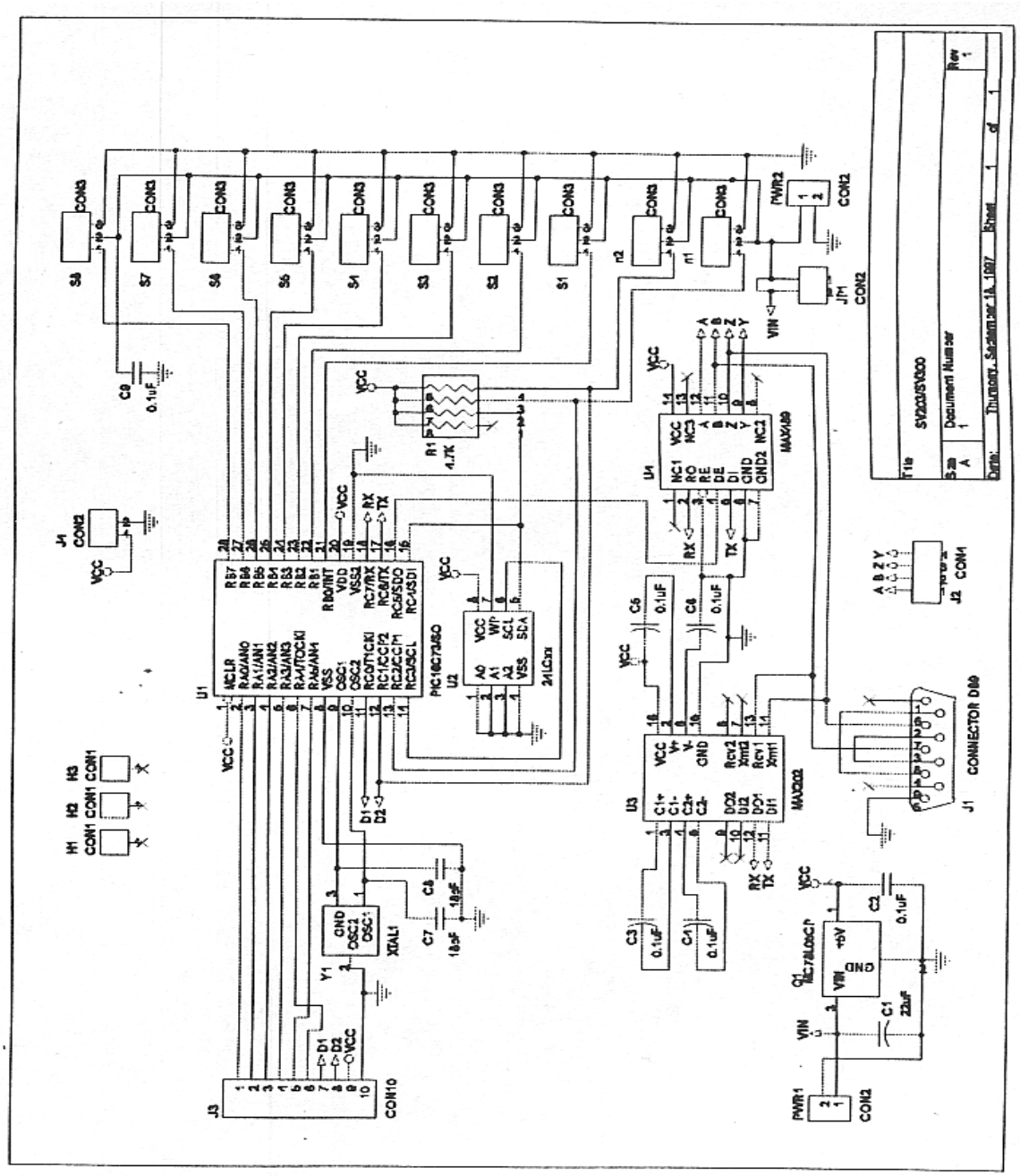
If the other boards need to be used for both input and output then use the wiring below. Because the RS232C standard was designed for a single peripheral, the output pin of the boards cannot be tied directly together.



SV203/300 - **Command Listing**

Commands	Parameter (n)	Description
BDn	0 to 255	Board Select
SVn	1 to 8	Servo Select
Mn	0 to 255	Move to an absolute location
In	-128 to 127	Move relative to current position
Dn	1 to 65535	Delay (ms)
PSn **PC**n **PT**n	1 to 8	Put, set single pin of servo port **b** is a number between 1 and 8, which tells what pin on the port is being used. If n = 0 -> clear Bit n = 1 -> Set Bit n = 2 -> Toggle Bit
ADn	1 to 5	Get A/D value, the board will return a value between "0" to "255" followed by ASCII 13 which represent a voltage between 0 to 5 Volts.
SOn	0 to 255	Shift a byte out to the SPI port
SI	None	Shift a byte in form the SPI port
WRm n	m = 0 to 255 n = 0 to 255	Write to internal RAM **m** is the memory location **n** is the value to write
RRm	m = 0 to 255	Read the contain of internal RAM **m** is the memory location to read
WEm n	m = 0 to 8190 n = 0 to 255	Write to external EEPROM **m** is the memory location **n** is the value to write
REm	m = 0 to 8190	Read the contents of external EEPROM **m** is the memory location to read
?	None	Help, return summary of command listing

SV203/300 - **Schematic**



Warranty

Pontech warrants its products against defects in materials and workmanship for a period of 90 days.

If you discover a defect, Pontech will, at its option, repair, replace, or refund the purchase price. Simply return the product with a description of the problem and a copy of your invoice (if you do not have your invoice, please include your name and telephone number).

The warranty does no apply if product has been damaged by accident, abuse, or misuse.

Copyright and Trademarks

Copyright(c) 1997 by Pontech. All rights reserved.
PIC is a registered trademark of Microchip Technology, Inc.
FUTABA is a registered trademark of FUTABA Corporations.

Disclaimer of Liability

Pontech is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of property, and any costs or recovering, reprogramming, or reproducing and data stored in or used with Pontech products.

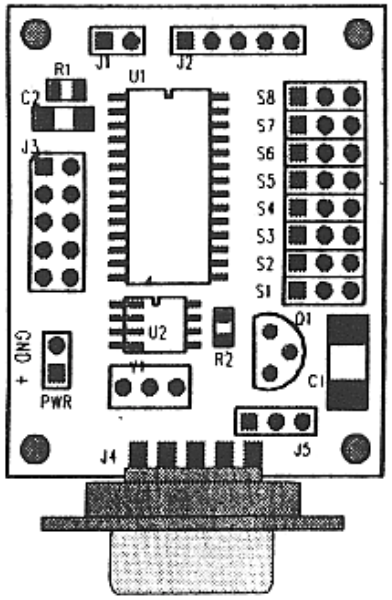
Jameco Part number 153648

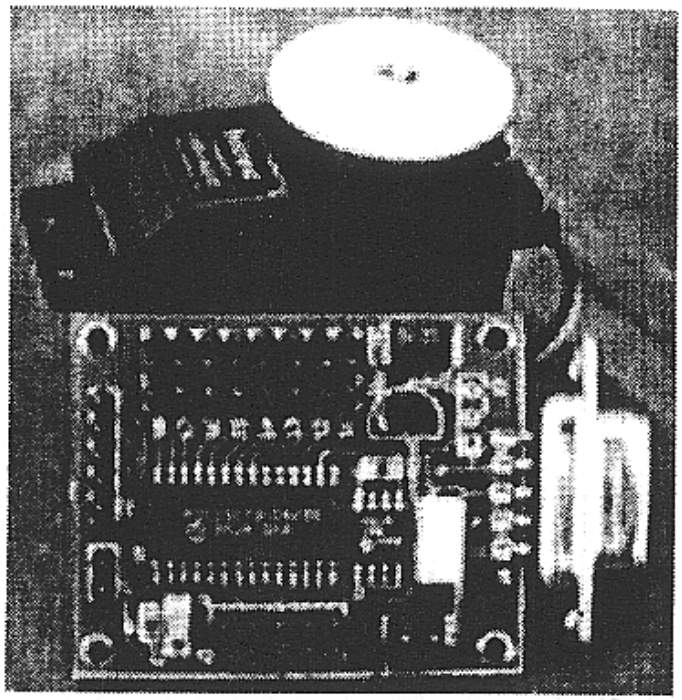
SV203B/C - **Table Of Contents**

TOPIC	**PAGE**
Servo Motor Controller Board	2
Feature List	3
Programming Process	4
Programming Language	6
Compile Limitations	8
Commands ASC, CALL, CONST	9
Commands DIM, DELAY	10
Commands DOLOOP, END	11
Commands EPEEK, EPOKE	12
Commands EXIT, FORNEXT	13
Commands GOTO, IFTHENELSE	14
Commands IRSEND, LET, PEEK, POKE	15
Commands PRINT, SPIIN, SPIOUT	16
Commands SUBEND SUB	17
Example Programs	18
Moving Servos	18
Using a Servo pin to control a relay	19
SPI Output	20
Modifying SV203B/C EEPROM Value	21
Control a PIC port, Modify PIC address	21
Reading A/D	22
SV203C IR Receiver Functions	23
SV203C IR Transmitter Functions	24
SVBAS Compiler Options	25
Appendix A - Command Summary	26
Appendix B - Memory Maps	27
Appendix B - IR Button Table	27
Appendix C - PIC16C73 AD Port	29
Appendix D - SVBAS.INC listing	30
Appendix E - ASCII Table	31

Jameco Part number 153648

SV203B/C - Servo Motor Controller Board





This is the supplemental manual for the SV203B/C servo motor controller. The SV203B/C is functionally equivalent to the SV203 with the added feature of being able to run a stand alone BASIC program stored on board the 8k EEPROM. Programs of approximately 2000 lines can be written and downloaded to the SV203B/C from an IBM compatible PC. The SV203C has the added ability of receiving infrared signals from a Sony TV remote or equivalent, which can be interpreted by the stand alone BASIC program, sent out a serial port, or both. In addition the SV203C can also transmit Sony TV remote codes. This can be used to control another SV203C or a Sony TV.

Jameco Part number 153648

SV203B/C - **Feature List**

SV203B/C

- Controls 1 to 8 servos per board, 8-bit precision.
- Servo port can be reconfigured for digital output to drive on/off devices. Source/Sink 25mA per pin.
- Interface to PC through RS232 port, 2400-19.2k bps (9600bps default).
- User definable board ID number, allowing multiple boards to share the same serial line.
- 5 Channel, 8 bit A/D input port for reading 0 - 5 Volts (control servo positions via Joystick/pot).
- An SPI port for shifting serial data in/out.
- 8k EEPROM for storing standalone routines or special movements (version B & C).
- IR-Receiver feature (version C).
- IR-Transmitter feature (version C).
- Servo Connectors: 3 pin sip, Futaba J-type connectors.
- Power supply: 4.8V to 6.0V.

Jameco Part number 153648

SV203B/C - **Programming Process**

In order for the controller board to run stand alone, sequences of commands must be saved into it's EEPROM (Electronically Erasable Programmable Read Only Memory). These commands are in the form of a basic program that is edited on a PC. Once a program is written it must be compiled and downloaded with the BASIC compiler (SVBAS.EXE) which outputs a hex object file that can then be downloaded to the SV203B/C. If there are no errors in the source code, the SVBAS program will automatically download the object code to the board.

C:> SVBAS <filename>

The program can then be run by typing:

C:> SVBAS /R or
C:> SVBAS /RUN

If you want the program to start running as soon as the board receives power then the Auto-start flag in the EEPROM must be set less than 255 or type the following command:

C:> SVBAS /AUTO

Once the on board BASIC program is running, it may be interrupted by typing:

C:> SVBAS /S or
C:> SVBAS /STOP

Jameco Part number 153648

SV203B/C - **Programming Process**

Here are the steps you might take to program the SV203B/C board.

1) First, use an editor to create a source code with extension .SV
   C:> EDIT TEST.SV

the contents of TEST.SV may look something like this

```
Dim I as Byte                 `Define one Byte
Main:                         `Label
  For I = 1 to 5              `Loop for 5 Times
    Servo1 = 10               `Move to position 10
    Delay 1000                `Delay for 1 sec.
    Servo1 = 100              `Move to position 100
    Delay 1000                `Delay for 1 sec.
  Next                        `Loop
End                           `Stop Running
```

2) Connect the serial port of the host PC to the board and supply power to the board.

3) To compile and download the source to COM2, type:

   C:> SVBAS Test /P2

If there is an error, then go to step 1 and fix the error. If there is not an error, the compiler will begin to download the program to the SV203B/C board.

To run the program on the SV203B/C connected to COM2 type:

   C:> SVBAS /R /P2

A Servo connected to S1 on the SV203B/C will move to position 10, delay for one second, then move to position 100 and delay for another second. The cycle will repeat for a total of five times and stop at position 100 about 10 seconds later.

SV203B/C - **Programming Language**

The SVBAB compiler uses a syntax loosely based on the QBASIC language by Microsoft (R). Because of the memory capacity of the EEPROM and limited power of the processor, not all the commands are implemented.

Naming Conventions

Functions and variables must be unique and begin with an Alpha character (a-z, A-Z) and may contain numeric characters and underscore. Other characters may work but are not supported. The following names are legal:

okay1
okay_1

The following name is illegal:

1toMany `Starts with a non-alpha char

Case Insensitive

The SVBAS compiler is not case sensitive for variables, constants, labels, function names or reserved words. A literal between quotes is, however, case sensitive. The following two code fragments do the same thing:

PRINT "Hello World" `Fragment one

print "Hello World" `Fragment two

Comments

Comments can be put on any line. Once a `character is encountered, the compiler treats the rest of the line as a comment.

SV203B/C - **Programming Language**

Colon Delimiter (Multiple commands on one line)

More than one command can exist on a line. The maximum length the SVBAS compiler can handle is 256 characters. Commands are separated with the: (colon) character.

```
; : PRINT "World"
```

Supported Operators

=	assignment
+	addition
-	subtraction

Bitwise (used during an assignment)
AND,OR,XOR,NOT

A = B AND 00001111b

Logical (used in an if statement expression)
=, <, <=, >, >=, <>, AND, OR, NOT

If A >= 5 AND B = 3 THEN 'Code fragment

Predefined variables

There are several predefined variables in the SVBAS.INC file. These variables are to make programming the built in functions and ports simpler. A complete listing of SVBAS.INC is in Appendix C.

Labels and Flow Control

Labels must start at the first colum of a line and must end with a colon. There can not be any space between the label name and the colon. When jumping to a label with a GOTO or other flow control statement, the colon must not follow the label name.

SV203B/C - **Programming Language**

Compiler Limitations

The compiler only allows 8-bit or 1-bit variables. Constants can be 16 bits but the only command that can use 16 bit constants is DELAY.

The SVBAS compiler does not support local variables in subroutines. All variables are global to all functions.

Subroutines do not support argument passing, use global variables instead.

RAM Space on the SV203B/C is limited to 60 bytes and is shared with the CALL and expression evaluation stack.

RAM Space on the SV203B/C also takes over most of the RS232 serial buffer used in SV203 non-standalone mode. This means the buffer is only 20 characters long and not 80 characters long in as in the stand alone mode.

Multiplication and division operators are not implemented.

SV203B/C - ASC, CALL, CONST

ASC

Return a byte value for an ASCII character. This function is useful for transmitting an IR value.

Syntax:
ASC <character>
character is an ASCII character encapsulated by double quotes

Example:
DIM ASCII AS BYTE
ASCII = ASC "A"

CALL

A statement that temporarily transfers control to a BASIC subroutine. When the subroutine ends with an END SUB or EXIT SUB, program flow will continue at the next line after the CALL statement.

Syntax:
CALL <subName>
subName is the name of the SUB

Example:
CALL SlowMove

CONST

A non-executable statement that declares a symbolic constant to use in place of numeric values. Symbols declared with a CONST are evaluated at compile time and do not require any RAM.

Syntax:
CONST <constName> = <value> constName is a name that follows SVBAS naming rules, value is a constant number.

Example:
CONST a = 5

SV203B/C - **DIM**

DIM

A declaration statement that names a variable and allocates storage space for it.

Syntax1:
DIM <variable> [AS BYTE]
variable is a BASIC variable name
all variables are shared for all procedures

Syntax2:
DIM <bitVariable> AS BIT <bitNumber> of <variable>
bitVariable is a BASIC variable name
all bit variables are shared for all procedures
bitNumber is a number between 0 and 7
variable is a declared variable
note: a variable declared as a bit can only be assigned to 0 or 1, False or True

Example:
DIM A AS BYTE Declare one byte
DIM bA2 AS BYTE 2 OF A Declare one bit

DELAY

Built in function that delays n milliseconds.

Syntax:
DELAY <variable I constant I literal>
variable is a BASIC byte variable name
constant and literal are two byte unsigned values (0 - 65535)

Example:
DIM A AS BYTE Declare one byte
A = 100
DELAY A pause for .1 second or
FOR A = 1 to 10 pause for 5 minutes (10 x 30sec)
    DELAY 30000 pause for 30 seconds
NEXT

SV203B/C - **DO...LOOP, END**

DO...LOOP

A flow control statement that repeats a block of statements while a condition is true or until a condition becomes true.

Syntax 1:

```
DO [{WHILE I UNTIL} expression]
        [statement block]

LOOP
```

Syntax 2:

```
DO
        [statement block]
LOOP [{WHILE I UNTIL} expression]
```

expression is a Boolean expression that will return non-zero (true), or zero (false) statement block is any number of statements on one or more lines which are to be executed as long as expression is true. WHILE continues to loop while expression is true. UNTIL continues to loop until expression is true.

Example:

```
DO      Loop forever
        PRINT 'ABC"
LOOP        Goto DO
```

END

A BASIC declaration that ends a programs execution.

Syntax:

```
END
```

SV203B/C - **EPEEK, EPOKE**

EPEEK

Read an SV203B/C EEPROM address.

Syntax:
EPEEK <address>
address is the EEPROM address in question, See memory map appendix for more information. (value from 0 to 8191)

Example:
PRINT EPEEK(1) Send the power-up position of servo 1 out the serial port

---

EPOKE
Modify an EEPROM value.

WARNING: Because of the limited ERASE/WRITE cycles of the EEPROM, excessive writes to the EEPROM can permanently damage the device.

Syntax:
EPOKE <address>, <value>
address is the EEPROM location to be modified (value from 0 to 8191)
value is the value to be stored at address

Example:
EPOKE 1, 200 Set the power position of servo 1 to 200

SV203B/C -EXIT, FOR...NEXT

EXIT

A flow control statement that exits a, DO...LOOP, FOR...NEXT loop, or SUB.

Syntax:
EXIT {DO I FOR I SUB}

Example:
```
DIM a AS byte
FOR a = 1 to 20
        IF a = 10 THEN EXIT FOR
        PRINT a
NEXT
```

FOR...NEXT

A flow control statement that repeats a block of statements a specified number of times.

Syntax:
```
FOR counter = start TO end [STEP increment]
        [statements]
NEXT
```
counter is a numeric variable used as the loop counter
start is the initial value of the counter
end is the final value of the counter
increment is the amount the counter is incremented each time through the loop

Example:
```
DIM a AS byte
FOR a = 50 to 1 STEP -2 Count down from 50 to 1 in steps on 2
        PRINT a
NEXT
```

Jameco Part number 153648

SV203B/C - GOTO, IF...ELSE...THEN

GOTO

A flow control statement that branches unconditionally to the specified line.

Syntax:
GOTO <label> label is the label of the line to execute next. This line must be in the same procedure or subroutine as the GOTO statement

Example:
```
start: PRINT "Hello World"
GOTO start
```

IF ... THEN ... ELSE

A flow control statement that allows conditional execution or branching, based on the evaluation of an expression that must be either true or false.

Syntax:

```
IF expression THEN
        [statement block]
[ELSEIF expression THEN
        [statement block]
...
[ELSE
        [statement block]
END IF
```

IF expression THEN part expression is a Boolean expression that must return non-zero (true) or zero (false) statement block consists of any number of statements on one or more lines.

Example:

```
IF a >= 10 THEN
        c = 20
ELSEIF b < 5 THEN
        c = 1
END IF
```

SV203B/C - IRSEND, LET, PEEK, POKE

IRSEND

Built in function that generates the Sony Remote 40kHz AM signal based on the value stored in IRreg, to be transmitted via an IR LED. Value can be 0 - 255

Example:
IRreg = ASC(F) Value for the POWER button CALL Irsend

LET

Assigns an expression to a variable.

Syntax:
[LET] <variable> = <expression I constant I literal>
Note: LET is optional and not required.

Example:
LET Servo1 = 128 Move Servo 1 to position 128
Servo1 = 128 Comparable to above

PEEK

Read a value stored in RAM.

Syntax:
PEEK <address> address is the RAM address in question, See memory map appendix for more information.

Example:

```
PRINT EPEEK(51) Send the current position of servo1 out
the serial port
```

POKE

Modify a RAM value.

Syntax:
POKE <address>, <value> address is the RAM location to be modified value is the value to be stored at address

Example:
POKE 1, 200 Set the current position of servo 1 to 200

SV203B/C - **PRINT, SPIIN, SPIOUT**

PRINT

A device output statement that outputs data to the RS232 serial port.

Syntax:
PRINT [expression-list][{,I;}]
If all arguments are omitted, a blank line is printed. If expression-list is included, the values of the expressions are printed on the screen.
The expressions in the list may be numeric or string expressions. (String literal must be enclosed in quotation marks.)
The optional characters "," and ";" can be use to send a tab or no return respectively at the end of the string.

Example:
PRINT Hello World!"
PRINT a

SPIIN

Built in function that shifts data from the SPI port into SPIreg.

Example:
CALL SPIin
PRINT SPIreg

SPIOUT

Built in function that shifts data from the SPIreg out the SPI port.

Example:
SPIreg = 55
CALL SPIout

SV203B/C - **SUB ... END SUB**

SUB...END SUB

A procedure statement that marks the beginning and end of a subprogram.

Syntax:

```
SUB name
        [statements]
        [EXIT SUB]
        [statements]
END SUB
```

Example:

```
SUB PrintHello
        Print Hello"
END SUB
```

SV203B/C - **Example Programs**

One of the most productive ways to learn about a programming language is to look at examples written by others, so here they are...

Moving Servos
This program will move servo 1 to servo 8 through positions 15 to 215 in steps of 50 with one-second delay between moves. The variables Servo1 through Servo8 are predefined in the file SVBAS.INC. By assigning a value from 0 to 255 to ServoX, you are affecting the PWM for that servo channel, 0 = No PWM, 1 = 0.6ms pulse width, and 255 = 2.4ms pulse width.

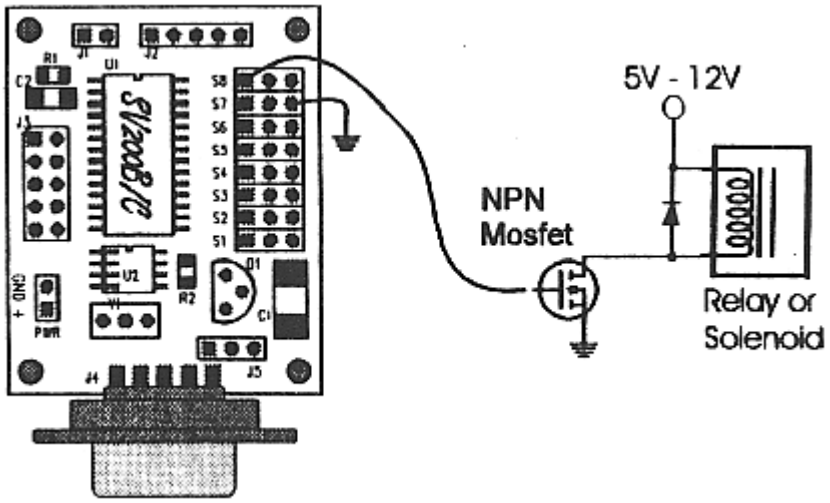
```
DIM position AS byte start:
FOR position = 15 TO 215 STEP 50
        Servo1 = position 'Servo1 - Servo8 are
        Servo2 = position 'predefined in the
        Servo3 = position 'SVBAS.INC file
        Servo4 = position
        Servo5 = position
        Servo6 = position
        Servo7 = position
        Servo8 = position
        DELAY 1000 'Delay for 1 second
NEXT
GOTO start
```

SV203B/C - **Example Programs**

Using a Servo pin to control a relay

This program reconfigures Servo8 output to switch a relay on and off every second.

```
`Port B is the SV203 servo PWM port,
`RB is defined in SVBAS.INC
DIM relay8 AS bit 7 of RB

Servo8 = 0 `Turn off Servo8 PWM
start:
        relay8 = 0     `Turn off relay
        DELAY 1000         `Delay for 1 second
        relay8 = 1     `Turn on relay
        DELAY 1000         `Delay for 1 second
GOTO start
```

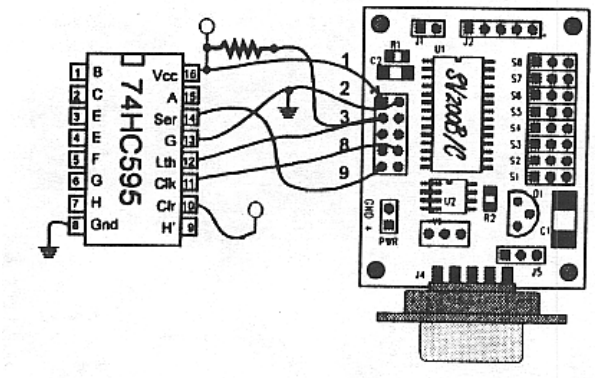


SV203B/C - **Example Programs**

SPI Output

This program requires a 74HC595 (serial to parallel shift register with output latch). The 74HC595 serial in and serial clock are connected to the SPI port and the latch clock is connected to RA4 of the SV203B/C. The program shifts out an alternating bit patteren and latches the value to the output of the 74HC595.

```
DIM bTRISA4 as Bit 4 of TRISA
DIM bSPIlatch as BIT 4 of RA

bTRISA4 = 0         'Set RA.4 as output
bSPIlatch = 1       'Set latch clock high
Main:
        SPIreg = 01010101b      'Set value to be shifted
        call SPIout   'Shift out the data
        bSPIlatch = 0       'pulse latch clock
        bSPIlatch = 1
        delay 1000   'Pause one second
        SPIreg = 10101010b      'Alternate the bit pattern
        call SPIout   'Shift out the data
        bSPIlatch = 0       'pulse latch clock
        bSPIlatch = 1
        delay 1000   'Pause one second
Goto Main
```



SV203B/C - **Example Programs**

Slow Moves

With the use of FOR loops and short delays, a slow move command can be implemented.

```
DIM I as byte : DIM Pos as byte'Pos is the new Const slowDelay = 5
        'position
Sub Slowl
        IF Servol > Pos THEN
                FOR I = Servol TO Pos    'Slowly increase
                        Servol = I : Delay slowDelay      'to Pos
                Next
        ELSE
                FOR I = Servol TO Pos Step -1  'Slowly decrease
                        Servol = I : Delay slowDelay      'to Pos
                Next
        END IF
END SUB
```

Modifying SV203B/C EEPROM Value

This simple program modifies EEPROM address 1 of the SV203B/C. This address determines the power-up position of servo 1.

```
'Check if value is already set to avoid excessive writes the EEPROM
IF EPEEK (1) <> 230 THEN EPOKE 1, 230
```

Control a PIC port/Modify PIC address

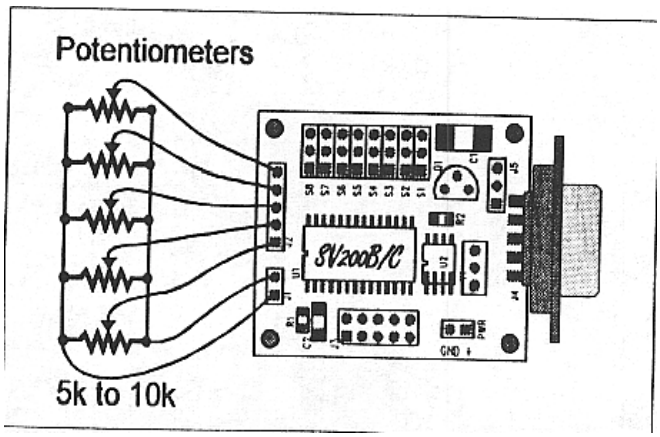
This simple program modifies the contents of internal RAM and demonstrates the equivalence of setting the position of Servo 1 using either RAM address 51 of the SV203B/C or the Servo 1 variable defined in the SVBAS.INC file.

```
POKE 51, 25 'Set Servo1 to position 25
PRINT PEEK (51) 'Send out current servo position
Servo1 = 25 'Identical to above PEEK, POKE example
PRINT Servo1
```

Reading A/D

This program reads A/D (Analog to Digital) channel 1 through channel 5 and then moves Servo1 through Servo5 respectively, to the position of the A/D value. Note that channels 1 to 5 are AD0 to AD4.

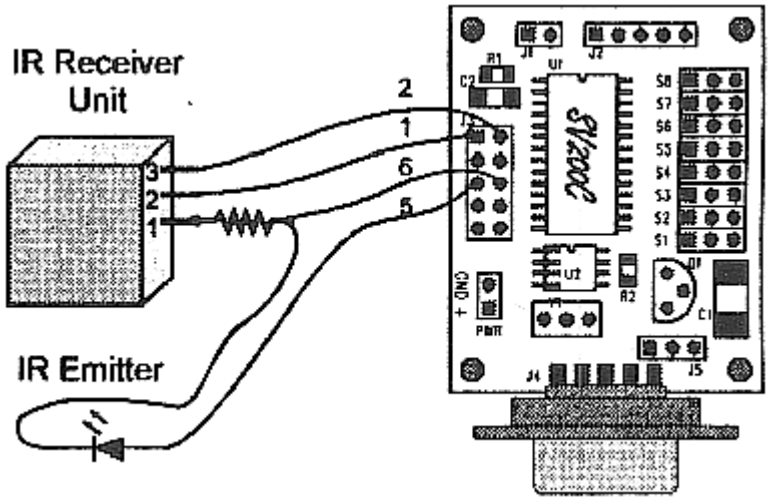
```
'ADRES, ASCON0 and ADCON1 are defined in SVBAS.INC

    ADCON1 = 0 'Set all A/D ports to Analog operation start:

ADCON0 = 11000001b          `Select AD0
bADSTART = 1                `Start Conversion
Servo1 = ADRES              `Assign result to Servo1
ADCON0 = 11001001b          `Select AD1
bADSTART = 1                `Start Conversion
Servo2 = ADRES              `Assign result to Servo2
ADCON0 = 11010001b          `Select AD2
bADSTART = 1                `Start Conversion
Servo3 = ADRES              `Assign result to Servo3
ADCON0 = 11011001b          `Select AD3
bADSTART = 1                `Start Conversion
Servo4 = ADRES              `Assign result to Servo4
ADCON0 = 11101001b          `Select AD4
bADSTART = 1                `Start Conversion
Servo5 = ADRES              `Assign result to Servo5
GOTO start
END
```



Jameco Part number 153648

SV203B/C - **SV203C IR Receiver Functions**

To use the infrared (IR) function, the IR receiver unit must be connected to port C or connector J3 of the SV203C board.



Any IR remote controller made for a Sony TV or any universal remote set to emulate a Sony TV remote will work with the SV203C. The SV203C will pick up most of the buttons that are on the remote. This includes the number keys 0 to 9, Enter, Power, Channel Up, Channel Down, Volume Up, Volume Down, TV/VCR and Muting Buttons.

The board must be programmed to detect the IR signal.

```
bIRenable = 1                    'Enable IR receiver
DO                               'Loop forever
  IF bIRreceive Then             'If IR received in Buffer
   PRINT IRreg                   'Send IR value out serial port
   BIRreceive = 0                'Clear flag for next receive
  END IF                         'End of IF statement
LOOP                             'Loop back to DO
```

This program will wait in an idle loop until an IR signal is received. The value received will be sent out the serial port and wait for another IR signal.

Jameco Part number 153648

SV203B/C - **SV203C IR Receiver Functions**

This program detects a number from 1 to 8 from an IR remote to select a servo. Once a number is pressed the channel up/down is used to move the servo in increments of three.

```
DIM SelectServo AS byte
BIRenable = 1                          `Enable IR receiver
DO                                     `Loop forever
 IF bIRreceive Then                    `If IR received in Buffer
  IF IRreg >= ASC ("1")                  and IRreg <= ASC ("9") then
   ServoSelect = IRreg                      `Save Selected Servo
  ELSE
   IF IRreg = ASC ("A")                then Call ChannelUP
   IF IRreg = ASC ("B")                then Call ChannelDOWN
  END IF
  bIRreceive = 0                       `Clear flag for next IRreceive
 END IF
LOOP                                   `Loop back to DO
        SUB ChannelUP
                IF Select Servo = ASC ("1") then Servo1 = Servo1 + 3
                IF SelectServo = ASC ("2") then Servo2 = Servo2 + 3
                IF SelectServo = ASC ("3") then Servo3 = Servo3 + 3
                IF SelectServo = ASC ("4") then Servo4 = Servo4 + 3
                IF SelectServo = ASC ("5") then Servo5 = Servo5
                + 3
                IF SelectServo = ASC ("6") then Servo6 = Servo6 + 3
                IF Select Servo = ASC ("7") then Servo7 = Servo7 + 3
                IF SelectServo = ASC ("8") then Servo8 = Servo8 + 3
        END SUB
        SUB ChannelDOWN
                IF SelectServo = ASC ("1") then Servo1 = Servo1 - 3
                IF SelectServo = ASC ("2") then Servo2 = Servo2 - 3
                IF SelectServo = ASC ("3") then Servo3 = Servo3 - 3
                IF SelectServo = ASC ("4") then Servo4 = Servo4 - 3
                IF SelectServo = ASC ("5") then Servo5 = Servo5 - 3
                IF SelectServo = ASC ("6") then Servo6 = Servo6 - 3
                IF SelectServo = ASC ("7") then Servo7 = Servo7 - 3
                IF SelectServo = ASC ("8") then Servo8 = Servo8 - 3
        END SUB
```

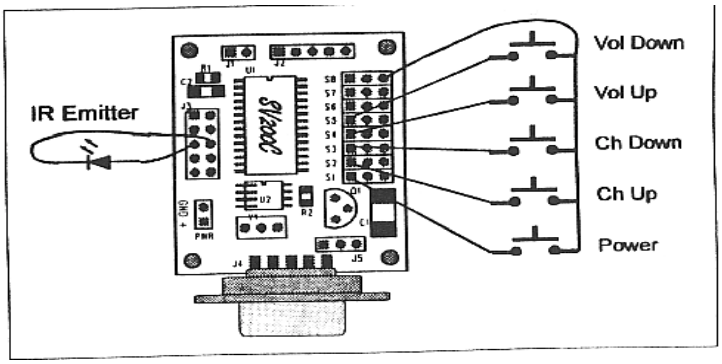
Jameco Part number 153648

SV203B/C - **SV203C IR Transmitter Functions**

The SV203C has an IR transmit feature that allows the board to send Sony style signals over an infrared emitting diode connected to pins 5 and 6 of port C.

This program reconfigures port B as a digital input. When a button is pressed, an IR signal corresponding to each button will be sent out the IR LED to control a SONY TV or another SV203C board remotely.

```
DIM bPWR as BIT 0 of RB
DIM bChUp as BIT 1 of RB
DIM bChDn as BIT 2 of RB
DIM bVolUp as BIT 3 of RB
DIM bVolDn as BIT 4 of RB
DIM OPTION at 129 'Register for port B pullup
DIM bRBPU as BIT 7 of OPTION

Servo1 = 0 : Servo2 = 0 : Servo3 = 0 : Servo4 = 0
Servo5 = 0 : Servo6 = 0 : Servo7 = 0 : Servo8 = 0
bRBPU = 0  'Enable weak Port B pull-up
TRISB = 11111111b       'Configure Port B as all inputs
DO

    if not bPWR then IRreg = ASC("F") : Call IRsend
    if not bChUp then IRreg = ASC("A") : Call IRsend
    if not bChDn then IRreg = ASC("B") : Call IRsend
    if not bVolUp then IRreg = ASC("C") : Call IRsend
    if not bVolDn then IRreg = ASC("D") : Call IRsend
    Delay 5 'Small delay for debounce
LOOP                    'Loop back to DO
```



SV203B/C - **SVBAS Compiler Options**

The BASIC compiler converts QBASIC like code into a PONTECH assembly. If there are no errors, it will assemble the code and download it to the board connected to the serial port.

SVBAS <filename> <options>

Command Line Options

/R or /RUN
Start program currently in EEPROM running.

/S or /STOP
Stop running program.

/P1 /P2 /P3 /P4 (default /P1 -> COM1).
Select COM port SV203B/C is connected to

/AUTO
Turn on Auto-Run (Start running from power-up)

/NOAUTO
Turn off Auto-Run (Don't run from power-up)

/C
Compile only (Generates assembly code)

/A
Assemble only (Generates hex object code)

/D
Download only (Downloads hex object code)

/L
Generate listing file of Assembly and Hex object.

SV203B/C - **Appendix B Memory Maps**

RAM Memory Map:

Address (m)	Description	Name
5	Port A	RA
6	Port B	RB
7	Port C	RC
30	A/D Result	ADRES
31	A/D Configuration 0	ADCON0
51	Servo 1 Position	SERVO1
52	Servo 2 Position	SERVO2
53	Servo 3 Position	SERVO3
54	Servo 4 Position	SERVO4
51	Servo 5 Position	SERVO5
52	Servo 6 Position	SERVO6
53	Servo 7 Position	SERVO7
54	Servo 8 Position	SERVO8
60	SPI Configuration	SPICON
61	SPI Register	SPIREG
62	IR Configuration	IRCON
63	IR Register	IRREG
133	Port A Direction	TRISA
134	Port C Direction	TRISB
135	Port C Direction	TRISC
159	A/D Configuration 1	ADCON1

Note: All other RAM locations not listed are used by the system and should not be used.

SV203B/C - **Appendix B Memory Maps**

EEPROM Memory Map:

Address (m)	Usage	Factory default	Note
0	Board ID #	1	
1	Initial Servo #1 Value	128	0 = off or Digital
2	Initial Servo #2 Value	128	0 = off or Digital
3	Initial Servo #3 Value	128	0 = off or Digital
4	Initial Servo #4 Value	128	0 = off or Digital
5	Initial Servo #5 Value	128	0 = off or Digital
6	Initial Servo #6 Value	128	0 = off or Digital
7	Initial Servo #7 Value	128	0 = off or Digital
8	Initial Servo #8 Value	128	0 = off or Digital
9	Baud Rate	50 (9600 baud)	25 (19200 baud) 100(4800 baud) 200(2400 baud)
10	Pre Enable Flag	1	1=Yes 0=No
11	Shift Configuration Register	0	MSB, valid on Rising, 8-bit
12-14	Reserved by PONTECH	Unknown	Unused
15	Auto-run at Power-Up	Unchanged	0-254 = auto-run 255 = no auto-run
16-63	Reserved by PONTECH	Unknown	Unused
64-8191	Program Space	Unknown	User Program

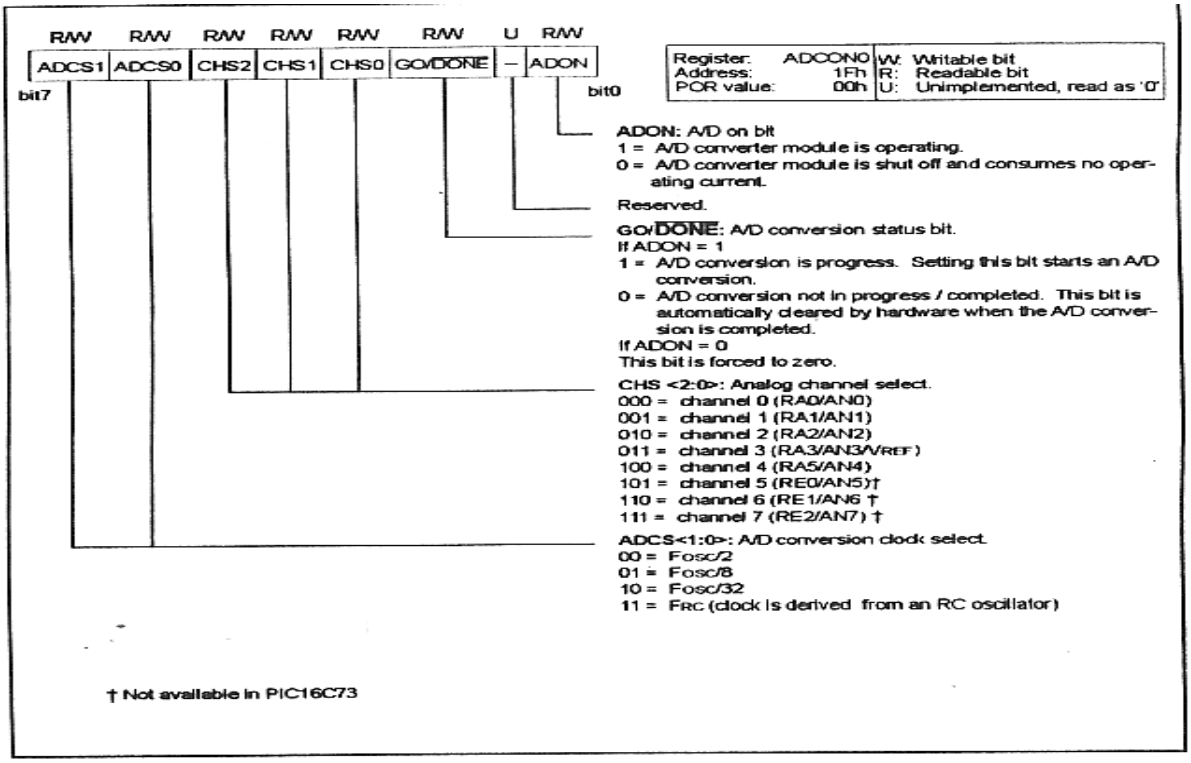
IR Button Table:

Button	ASC()	Decimal
1 - 9	"1" - "9"	49 - 57
0	":"	58
Enter	"<"	60
Channel Up	"A"	65
Channel Down	"B"	66
Volume Up	"C"	67
Volume Down	"D"	68
Muting	"E"	69
Power	"F"	70
TV/VCR	"V"	86

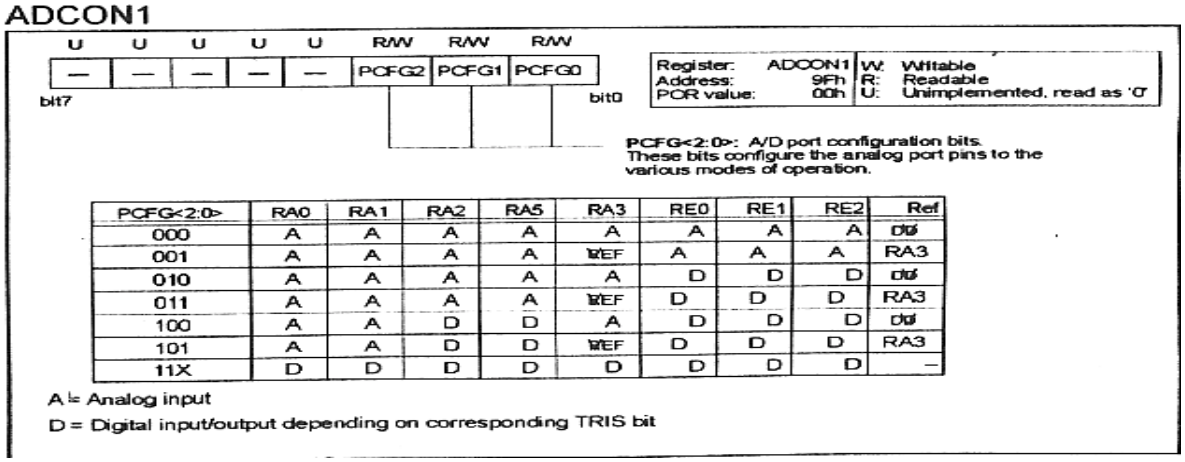
Jameco Part number 153648

SV203B/C - **Appendix C PIC16C73 A/D Port**

ADCON0


R/W R/W R/W R/W R/W R/W U R/W
ADCS1 ADCS0 CHS2 CHS1 CHS0 GO/DONE – ADON
bit7
bit0
Register: ADCON0
Address: 1Fh
POR value: 00h
W: Writable bit
R: Readable bit
U: Unimplemented, read as '0'
ADON: A/D on bit
1 = A/D converter module is operating.
0 = A/D converter module is shut off and consumes no operating current.
Reserved.
GO/DONE: A/D conversion status bit.
If ADON = 1
1 = A/D conversion is progress. Setting this bit starts an A/D conversion.
0 = A/D conversion not in progress / completed. This bit is automatically cleared by hardware when the A/D conversion is completed.
If ADON = 0
This bit is forced to zero.
CHS <2:0>: Analog channel select.
000 = channel 0 (RA0/AN0)
001 = channel 1 (RA1/AN1)
010 = channel 2 (RA2/AN2)
011 = channel 3 (RA3/AN3/VREF)
100 = channel 4 (RA5/AN4)
101 = channel 5 (RE0/AN5)†
110 = channel 6 (RE1/AN6 †
111 = channel 7 (RE2/AN7) †
ADCS<1:0>: A/D conversion clock select.
00 = Fosc/2
01 = Fosc/8
10 = Fosc/32
11 = FRC (clock is derived from an RC oscillator)
† Not available in PIC16C73

ADCON1


U U U U U R/W R/W R/W
— — — — — PCFG2 PCFG1 PCFG0
bit7
bit0
Register: ADCON1
Address: 9Fh
POR value: 00h
W: Writable
R: Readable
U: Unimplemented, read as '0'
PCFG<2:0>: A/D port configuration bits.
These bits configure the analog port pins to the various modes of operation.

PCFG<2:0>	RA0	RA1	RA2	RA5	RA3	RE0	RE1	RE2	Ref
000	A	A	A	A	A	A	A	A	VDD
001	A	A	A	A	VREF	A	A	A	RA3
010	A	A	A	A	A	D	D	D	VDD
011	A	A	A	A	VREF	D	D	D	RA3
100	A	A	D	D	A	D	D	D	VDD
101	A	A	D	D	VREF	D	D	D	RA3
11X	D	D	D	D	D	D	D	D	–

A = Analog input
D = Digital input/output depending on corresponding TRIS bit

Jameco Part number 153648

SV203B/C - **Appendix D SVBAS.INC**

```
***** Include file for SV203B/C *****
DIM     RA          AT      05h     'Port A
DIM     RB          AT      06h     'Port B
DIM     RC          AT      07h     'Port C

DIM     TRISA       AT      85h     'Port A Direction
DIM     TRISB       AT      86h     'Port B Direction
DIM     TRISC       AT      87h     'Port C Direction

DIM     ADRES       AT      1Eh     'A/D register
DIM     ADCONO      AT      1Fh     'A/D configuration 0
DIM     ADCON1      AT      9Fh     'A/D configuration 1
DIM     bADSTART as BIT 2 of ADCONO 'A/D start

'Interrupt Enable for IR feature
DIM     PIE2        AT      8Dh
DIM     STAT        AT      32h     'Status
DIM     bCARRY as BIT 0 of STAT
DIM     bZERO as BIT 2 of STAT
DIM     STACK       AT      27h     'Stack Pointer
DIM     SERVO1      AT      51      'Servo 1 position
DIM     SERVO2      AT      52      'Servo 2 position
DIM     SERVO3      AT      53      'Servo 3 position
DIM     SERVO4      AT      54      'Servo 4 position
DIM     SERVO5      AT      55      'Servo 5 position
DIM     SERVO6      AT      56      'Servo 6 position
DIM     SERVO7      AT      57      'Servo 7 position
DIM     SERVO8      AT      58      'Servo 8 position

DIM     SPICON      AT      60      'SPI configuration
DIM     SPIREG      AT      61      'SPI register

DIM     IRCON       AT      62      'IR configuration
DIM     IRREG       AT      63      'IR register
'Enable IR Flag
DIM     bIRenable as BIT 0 of PIE2
'IR valid and received Flag
DIM     bIRreceive as BIT 5 of IRCON
```

SV203B/C - **Appendix E ASCII Table**

000 (nul)	016 (dle)	032 (sp)	048**0**	064 @	080**P**	096	112**p**
001 (soh)	017 (dcl)	033 !	049**1**	065**A**	081**Q**	097**a**	113**q**
002 (stx)	018 (dc2)	034 "	050**2**	066**B**	082**R**	098**b**	114**r**
003 (etx)	019 (dc3)	035 #	051**3**	067**C**	083**S**	099**c**	115**s**
004 (eot)	020 (dc4)	036 $	052**4**	068**D**	084**T**	100**d**	116**t**
005 (enq)	021 (nak)	037 %	053**5**	069**E**	085**U**	101**e**	117**u**
006 (ack)	022 (syn)	038 &	054**6**	070**F**	086**V**	102**f**	118**v**
007 (bel)	023 (etb)	039 '	055**7**	071**G**	087**W**	103**g**	119**w**
008 (bs)	024 (can)	040 (	056**8**	072**H**	088**X**	104**h**	120**x**
009 (tab)	025 (em)	041)	057**9**	073**I**	089**Y**	105**I**	121**y**
010 (lf)	026 (eof)	042 *	058 :	074**J**	090**Z**	106**j**	122**z**
011 (vt)	027 (esc)	043 +	059 ;	075**K**	091 [	107**k**	123 {
012 (np)	028 (fs)	044,	060 <	076**L**	092 \	108**l**	124 |
013 (cr)	029 (gs)	045 -	061 =	077**M**	093]	109**m**	125 }
014 (so)	030 (rs)	046 .	062 >	078**N**	094 ^	110**n**	126 ~
015 (si)	031 (us)	047 /	063 ?	079**O**	095 _	111**o**	127 (del)